Fault localization in reversible circuits is easier than for classical circuits

Kavitha Ramasamy, Radhika Tagare, Edward Perkins and Marek Perkowski
Department of Electrical and Computer Engineering, Portland State University, Portland, Oregon, 97207-0751, mperkows@ee.pdx.edu

Abstract

There is recently an interest in test generation for reversible circuits, but nothing has been published about fault localization in such circuits. This paper deals with fault localization for binary reversible (permutative) circuits. We concentrate on functional test based fault localization, to detect and locate "stuck-at" faults in a reversible circuit by creating an adaptive tree. A striking property of reversible circuits is that they exhibit "symmetric" adaptive trees. This helps considerably by being able to generate only half of the tree, and the other half is created as the mirror image of the first half. Because each test covers half faults [1] and the fault table has a high density of ones, it is relatively easy to generate the tree. The problem of fault localization of reversible circuits is therefore easier than the same problem for standard irreversible circuits. We present some preliminary results from an approach using traditional adaptive tree methods. We propose also a new efficient algorithm that eliminates the fault table generation and dynamically creates the adaptive fault tree.

1. Introduction

Low power consumption is a major issue in VLSI circuits today. As the transistor size decreases, power consumption and heat dissipation become two major problems for the IC designers. Techniques like voltage scaling, low power layout are already in practice to obtain circuits with low power. The motivation for studying reversible adiabatic circuits comes mainly from this increasing demand for low energy dissipation computation [5]. Reversible circuits play also a vital role in quantum computing [6] and emerging nanotechnologies [7, 8, 9]. To ensure the proper functionality and the durability of an integrated circuit; testing and failure analysis are extremely important during and after its design and manufacturing. The main idea behind fault localization in future highly parallel redundant logic systems is self-repair based on localization and replacement of faulty modules. We show that this task is easier when the circuit is reversible. Fault localization can be used for process diagnosis, and for manual or automatic repair process. In automatic process a circuit with redundant modules is prefabricated, faults are localized and the circuit is reconfigured by replacing faulty modules by correct spare modules.

2. Test generation & fault localization

There are two aspects to testing a circuit. One is Fault Detection and the other is termed Fault Localization. Earlier involves the detection of *presence of fault* in a circuit; while the latter is about *finding the exact location of this fault*. So far, nothing has been published on self-repair and fault-localization of reversible circuits, although it is a common agreement that future technologies will be both low power and fault-tolerant.

A (deterministic) Fault Table [3] has all tests as rows and all faults as columns. A "1" at the intersection of row r_i and column c_j determines that test r_i detects fault c_j . In a non-deterministic Fault Table used to localize faults in quantum circuits [12] in addition to entries 1 meaning probability 1 there exist other probabilities of covering columns by rows.

In our approach we focus on functional test based fault localization to locate single stuck-at faults in binary reversible circuits. Our approach is based on the following propositions:

PROPOSITION 1. *If the complete fault table is first found then the complexity of the ordered adaptive (decision) tree created from this table does not depend upon the order of the variables (i.e. tests) An ordered tree can be created, i.e. one in which all variables from root to leafs are in the same order.* Since each test covers half of the faults, the choice of the root test is immaterial; the tree that is then derived from the root will locate all the faults.

PROPOSITION 2. *If the complete fault table is first found then it is sufficient to design ordered adaptive trees and there is no need to investigate free adaptive trees.* Observe that in classical logic the adaptive trees are free (order-less) to minimize the number of nodes in them. In adaptive trees for reversible logic circuits the number of nodes is minimized with tests ordered the same way in each branch.

PROPOSITION 3. *If the complete fault table is found, then only a half of the adaptive tree needs to be created based on recursive splitting sets of faults to subsets for each test. The other half of the tree can be constructed based on the symmetry property of reversible circuits.* This property shortens the tree generation process by half.

3. Previous work on testing binary reversible circuits

3.1. Test generation for reversible circuits and design for test of reversible circuits

Patel et al., [1] use a direct approach to generate set of test vectors to detect all faults in a reversible logic circuit by decomposing larger circuits into smaller sub-circuits (block partitioning). They formulate finding the minimal test set as an Integer Linear Programming (ILP) problem. Single stuck-at fault model is used to detect faults in internal lines and primary input and output lines of the circuit. Their main contributions are the following observations regarding reversible circuits:

i) Any test set that is complete for the single stuck-at fault model is also complete for multiple stuck-at fault model.
ii) Each test vector covers exactly half of the faults, and each fault is covered by exactly half of the possible test vectors.

Ugur Kalay et al., [2] use universal test set to detect faults in AND-EXOR based circuits. They too use a similar fault model as Patel et al. This method can be adopted for a special type of reversible cascades that are based on ESOP circuits [4]. Both [1] and [2] focus only on fault detection. Because of importance of fault localization, we extend these works towards this new aspect of reversible circuits.

3.2. Fault localization of irreversible circuits

The two most popular approaches to Fault Localization are *A) Preset Method* and *B) Adaptive Tree Method* [3]. In literature, both methods start from a fault table which has tests (input vectors) as rows and all possible faults as columns. The goal of preset method is to find a minimal test set to locate all the faults in the circuit. The minimal test set is found using the algorithm to solve the special covering problem in the so-called fault location table created from the fault table [3]. Size of this table limits the applicability of this approach to relatively small problems. Therefore we concentrate here only on the Adaptive Tree Method.

The Adaptive Tree Method: Adaptive tree is represented by a directed tree data structure. Observe that in case of non-reversible circuits, the tree is created based on the complete fault table. At each level of recursive tree generation, an attempt is made to choose a row in this table; that approximately covers half of the faults remaining in this node. This requires first to create the table, and next, to select a good row. Unfortunately, such a selection can be difficult since there are many candidates and often none is close to covering half faults. In reversible circuits, because of the property by *Patel et al* [1] that every test covers half faults, the adaptive tree generation is easier.

4. Adaptive tree generation for reversible circuits

We focus here on functional test based Fault Localization to locate single stuck-at faults in binary reversible circuits which particularly comprise of Toffoli, Feynman, Fredkin and NOT gates. We assume circuits with faults only in internal wires, primary input wires and output wires of the circuit. The circuit is analyzed by partitions (Pi) with one gate per partition. Partitions in our circuit (unlike Patel et al.) are only to locate nodes; we refer to each particular node (N) as (Pn, Nn) in the circuit as shown in Figure 1.

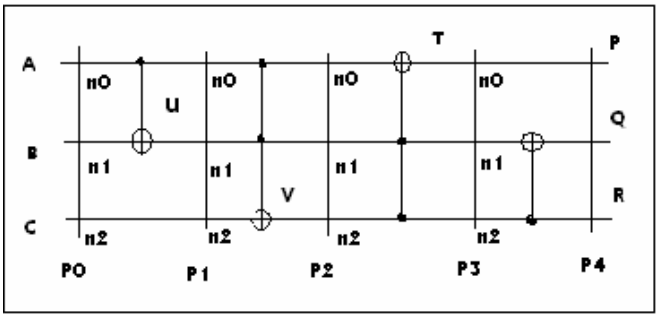

Figure 1: Example of a binary reversible circuit of width 3 and using feynman and toffoli gates

In Figure 1, A, B and C are the basic input wires. P0, P1, P2, P3, P4 are column cuts (or what we call partitions) such that only one gate is covered by each. P, Q, R are the circuit's primary outputs.

One of our current variants of fault localization is based on creating an adaptive tree by generating a standard complete and static fault table, shown in Table 1. The stuck-at circuit faults for each node A-R, after removing equivalent faults, were labeled f1 through f14. Column f0 is for the fault-free (FF) circuit.

Figure 2 shows an example of an Adaptive Tree created for the binary reversible circuit in Figure 1. The left branch of each node (good) is for a circuit passing the respective test and the right branch (fail) is for the circuit failing this test.

The choice of the test which splits faults at each node is based on this rule: *at every node choose a test that*

partitions the incoming subset/set of faults into the balanced subsets of faults (i.e. with their cardinalities as close as possible). (Such a choice creates a nearly well-balanced tree, thus the tree allows for the fastest fault localization assuming equal faults probabilities). Observe local lack of symmetry in the left bottom node which leads to a good circuit. Besides, every node (test) splits faults into two equal size sets or sets that sizes differ just by one.

Table 1 - Faults for example circuit

		S-A-0							S-A-1						
Test	FF	A	B	C	U	P	Q	R	A	B	C	U	P	Q	R
vector	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	0	0	1	1	1	1	0	1	1	0	0
2	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1
3	0	0	1	1	1	1	0	1	1	0	0	0	0	1	0
4	0	1	0	0	1	0	0	1	0	1	1	0	1	1	0
5	0	1	0	1	1	1	1	0	0	1	0	0	0	0	1
6	0	1	1	0	0	1	0	0	0	0	1	1	0	1	1
7	0	1	1	1	0	1	1	1	0	0	0	1	0	0	0

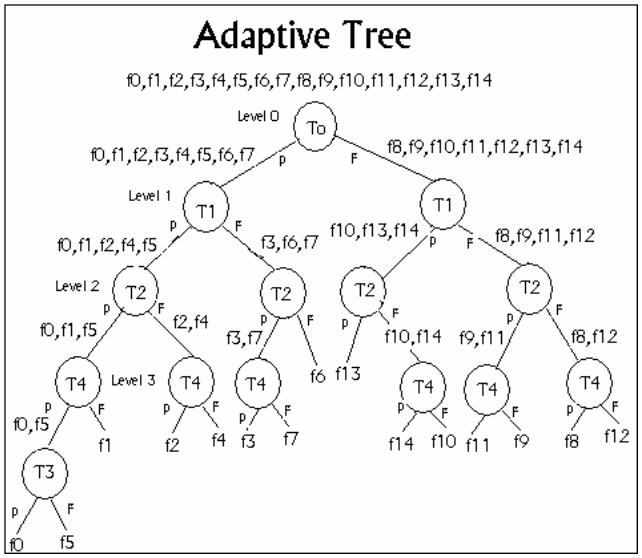

Figure 2: Example of an adaptive tree for a binary reversible circuit

Due to the symmetry property observed in reversible circuits, adaptive trees for reversible circuits, in any variant exhibit a special mirror image property when folded over the test at level 0. The tree is said to be symmetric because for a particular level, particular node, the same test vector can be used for splitting incoming faults at that node and the node in its mirror image.

4.1. Results of a variant in which adaptive tree was created from a complete fault table

We developed a program to perform fault localization using the traditional Adaptive Tree approach which incorporates a simulator to evaluate binary reversible circuits. The program generates a complete Fault Table for all internal nodes and outputs of a circuit. We tested this method on several circuits. Our simulator supported up to 3-input gates and constant zero inputs.

We used benchmarks from Maslov [11] plus our own test circuits (Small, Adapt). The circuits had 3 or 5 wires (a quantum width of 5). Benchmark Mod5 has one constant 0 input. The results are shown in Table 2. *Wr* stands for a number of wires (width), *Cn* for number of constants, *Lv* is a number of levels of the tree, *Nd* is the number of nodes and *EF* is the number of equivalent faults. Equivalent faults can be functional equivalence or wire equivalence. Equivalent faults are detected by comparing each fault column with all other column entries in the fault table. *Tests* is the number of test vectors based on 2^{Wr}.

All trees were symmetric, and were 4-5 levels deep. Circuit 3_17 was well-balanced. Others were nearly well-balanced (like the one from Figure 2). Processing time for each circuit (including both table generation and tree creation) was only a few seconds.

Table 2 - Adaptive tree fault localization results

Circuit	Gates	Wr	Cn	Tests	Lv	Nd	EF	Bal
Small	3	3	0	8	5	10	14	N
Adapt	4	5	0	32	5	18	32	N
3_17	6	3	0	8	4	16	26	Y
Mod5	9	5	1	32	5	24	76	N
Xor5	4	5	0	32	5	18	32	N

5. Proposed new greedy direct tree generation algorithm

To speed up the adaptive tree method and allow it to be used without generating all tests, we propose a new algorithm that can be applied to larger circuits. Our new algorithm for fault localization uses a novel approach to choose an input test vector based on the outputs of the intermediate nodes of the partition or column cut which has the maximum uncovered faults.

A counter test vector at this chosen partition is applied to find the corresponding test vector at the inputs, and then the test vector is checked for its coverage measure. An (incomplete, dynamic) fault table corresponding only to the tests created from the counter test vectors is generated dynamically while creating the adaptive tree. Our goal is to use this approach to locate both the unique and equivalent

faults in reversible circuits, while generating a symmetric adaptive tree. With this new algorithm, adaptive trees can be efficiently created for large circuits.

An advantage of our proposed algorithm is that it avoids creation of the entire fault table for every test vector and all possible faults. This saves time and memory space over the traditional approach, where all the faults need to be simulated for all input test vectors, regardless of the circuit output. Fewer test vectors need to be generated to the circuit with our approach.

5.1 Reversible simulator for fault localization

We designed a simulator to analyze binary reversible circuits and fault-simulate them. The simulator produces the circuit output at each node for a particular input test vector applied to it. This is done for a fault-free circuit and for this circuit with every possible fault inserted. The simulator reads in the circuit description and creates a set of partitions/column cuts. It can operate in either forward or back directions. It operates in forward direction to find the output of the circuit at every node of every partition/column cut in the circuit. Also it operates in backward direction to find the input test vector corresponding to a counter-test vector applied at some partition of the circuit. Note again, that it is the property of reversible circuits which allows for easy simulation of a circuit in both directions and in exactly the same way, because for each n-bit reversible cell F, there exist a unique inverse cell $F^{-1.}$ Moreover, the Feynman, Fredkin and Toffoli gates that we use are their own inverses which speeds-up the simulation further. Observe, that this method cannot be used for classical circuits because for irreversible circuits back simulation of faults requires backtracking and for reversible circuits it is a one-run procedure without backtracking. It cannot be also used for quantum circuits because back simulation may lead to non-pure primary input states.

Concluding on fault simulation. Because of reversibility properties, it is easier for reversible circuits than for quantum and classical circuits. It is also simpler than for the general-purpose quantum circuits. Such circuits are described by compositions of matrix products and Kronecker products of unitary matrices. The faults in them are modeled by inserting complex (unitary or not) matrices of faults and decoherence-related measurements (density matrices) in respective qubits in partition columns [12].

5.2 Efficient algorithm for fault localization

1) A fault coverage table is created and updated incrementally together with generating the adaptive tree. It is built as the tree is expanded from the root to the leafs by adding new nodes (this is a free tree, variables are not ordered). For every level we update this fault table to represent only the remaining uncovered faults.

2) For all Levels ahead : we use the following recursive procedure -

   a) Select a partition for which the fault table shows the maximum number of uncovered faults and mark that partition as checked.
   b) Find the counter test vector at that partition.
   c) Backtrack in the circuit from this point using the back simulator function, to find the corresponding input test vector. This input test vector when applied to the circuit will cover the uncovered faults at that particular partition.
   d) Apply this input test vector to the given circuit to find the output using the forward simulator function.
   e) Get the simulation output table for this vector.
   f) Check if this input test vector divides the uncovered faults (looking at the fault table) from the previous level into half.
   g) If so, the test is good. Then check if the same test holds good for all other nodes in the same level.
      - if not then discard the test. Go to step h)
      - if good go to step i)
   h) If the test is not good,
      - If all partitions are not checked, then choose a partition which is next maximum and repeat step b) onwards.
      - If all partitions are checked, then choose the input test vector which divides the uncovered faults into nearly half subsets.
      - Update the fault table by marking the covered faults by this chosen test vector. Go to step i)

   i) Repeat steps a) through i) until each leaf of the tree ends up with one or more non-separable stuck-at fault(s). All distinct single stuck-at faults except one, from the fault table are covered for every node in that level (since each leaf of the tree ends up with one and only fault)

5.3 Considering equivalent faults

When the output at two or more nodes of the circuit is identical for all input test vectors applied to the circuit, then those nodes are said to be equivalent nodes. In our proposed algorithm we deal with equivalent nodes which are adjacent to each other (in the same wire). In other words these nodes are nothing but a wire separated by the logical partitions P0, P1, etc. For example, in the given circuit in Figure 1, nodes n0 at partitions P0 and P1 are the equivalent nodes. In our incremental fault table we represent the equivalent nodes.

While considering the uncovered faults in a partition, we count the equivalent faults too; if uncovered until the moment. Other equivalent faults which are non-adjacent remain in the tree as non-separable sets of faults in the leafs.

5.4 Particulars to be noted

1) If there are N wires, then 2^N is the maximum possible length of the input test vector. We restrict ourselves to a certain number of test vectors as we choose a test vector depending on choice of outputs at one of the P partitions. Thus, the number of actual possible input test vectors for a particular circuit will depend on the number of partitions. Hence we can actually use only 2^P input test vectors.
2) The major consequence is that while doing so we might loose on some good test vectors at a particular node, which exactly divide the faults into two equal subsets of covered and uncovered faults.
3) Also another consequence is that we might loose on some good test vectors at a particular level, where the same test vector can be applied at every node in that level.
4) The effect of all these is that the adaptive tree will not be balanced and will not be ordered.

6. Future work

1) It is assumed in the algorithm 5.2. that all stuck-at-one faults are covered by a test vector which is all zeros; denoted by T0. But this holds true under the assumption that the circuit under test does not include any NOT gates i.e. inverters.

2) We would like to compare the speed of the algorithm from section 4.1 and the algorithm 5.2 to find how much the new approach is faster. This should be checked especially for large circuits. One reason to the efficiency is that the algorithm avoids creation of full table and search of all possible input test vectors.

3) In future, we also plan to modify our algorithm to incorporate fault localization in binary and multiple-valued quantum circuits [12]. Although the fault table and adaptive tree become in general deterministic or probabilistic, similar techniques to those presented in this paper can be used.

4) We need to develop a heuristic to handle the case of unseparable faults at leaf nodes. These faults may be able to be separated by choosing a different test vector sequence, or they may be truly equivalent. We have to develop a heuristic to handle all the equivalent faults.

7. Conclusions

We proved experimentally that propositions 1 – 3 are true when a complete Fault Table is created, thus fault localization in reversible circuits is simpler than in standard (non-reversible) circuits. This gives an initiative to redesign non-reversible circuits to make them more reversible (for instance by partitioning to reversible circuits and other circuits). So far, the motivation to perform research in reversible circuits was quantum realizations and low power design. High testability of these circuits gives a new motivation – even if we are dealing with standard CMOS circuit and adiabatic CMOS and our goal is not to save power, the reversible circuit design guarantees higher testability and fault localization. We dispose several examples of practical circuits that have been modified to (partially) reversible circuits and made thus more testable and fault-localizable. This is a forthcoming research area.

The program was tested on several reversible circuits from the literature [10]. Because of the lack of large benchmarks, we have to create many of these circuits randomly or we used circuits with no guarantee of their minimality. This is perhaps not a good idea, but nothing better can be done since there are no good synthesizers so far for very large reversible functions. We analyzed several examples with the width not more than 8. The symmetric property holds true for all them. The method is applicable to any kind of binary and multiple-valued reversible circuits; when the adaptive tree is created using standard adaptive tree approach with complete Fault Tables. The tree obtained is balanced or nearly balanced in these cases. But the tree may not be always symmetric for the reversible circuits that have NOT gates, when the tree is built using algorithm 5.2. Further testing and analysis of data is necessary on larger examples for both approaches. The method has been extended to multiple-valued reversible logic and binary and multiple-valued quantum circuits [12].

8. Acknowledgement

We would like to thank anonymous reviewers of this paper for their helpful suggestions and comments.

9. References

[1] K.N. Patel, J.P. Hayes and I. Markov, “Fault testing for reversible circuits,” Proc. *VLSI Test Symp.* (*VTS 03*), Napa, CA, pp. 410–416, April 2003

[2] U. Kalay, N. Venkataramaiah, A. Mishchenko, D. V. Hall, and M. A. Perkowski, “Highly Testable Finite State Machines Based on EXOR Logic”, *PACRIM'99 7th IEEE Pacific Rim Conference on communications, Computer and Signal Processing , Victoria, B.C. , Canada, August 23-25, 1999*

[3] Z. Kohavi, “Switching and Finite Automata Theory”, *McGraw Hill*, 1978.
[4] A. Mishchenko and M. Perkowski**,** ``Fast Heuristic Minimization of Exclusive Sums-of-Products," *Proc. RM'2001 Workshop*, August 2001
[5] C. Bennett, “Logic Reversibility of Computation,” *IBM J. Res. Dev.* 17:525-532, 1973.
[6] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.
[7] R. C. Merkle. Reversible electronic logic using switches. *Nanotechnology*, 4: pp. 21-40, 1993.
[8] R. C. Merkle. Two types of mechanical reversible logic. *Nanotechnology*, 4: pp. 114-131,1993.
[9] R. C. Merkle and K. E. Drexler. Helical logic. *Nanotechnology*, 7: pp. 325-339, 1996.
[10] D. Maslov, Reversible Logic Synthesis, *Ph.D. Thesis*, University of New Brunswick, 2003.
[11] D.Maslov, web site http://www.cs.uvic.ca/~dmaslov/ reversible benchmark circuits.
[12] S. Aligala, S. Ratakonda, K. Narayan, K.i Nagarajan, M. Lukac, J. Biamonte and M. Perkowski, Deterministic and Probabilistic Test Generation for Binary and Ternary Quantum Circuits, *Proceedings ULSI* 2004.
[13] R. Aitken, Test-Based Fault Localization – Part 1: Fault Models. *Agilent Technologies*, Santa Clara, California.
[14] S. Mitra, P.P. Shirvani, and E.J. Mc. Cluskey, Fault Location in FPGA-Based Reconfigurable Systems, http://crc.stanford.edu/crc_papers/mitrahldvt98.pdf
[15] I. Pomeranz, S.M. Reddy, Fault Location Based on Circuit Partitioning, *Proc. 1996 Intern. Conf. on Computer Design, ICCD’96*, Austin, Texas, p.154.